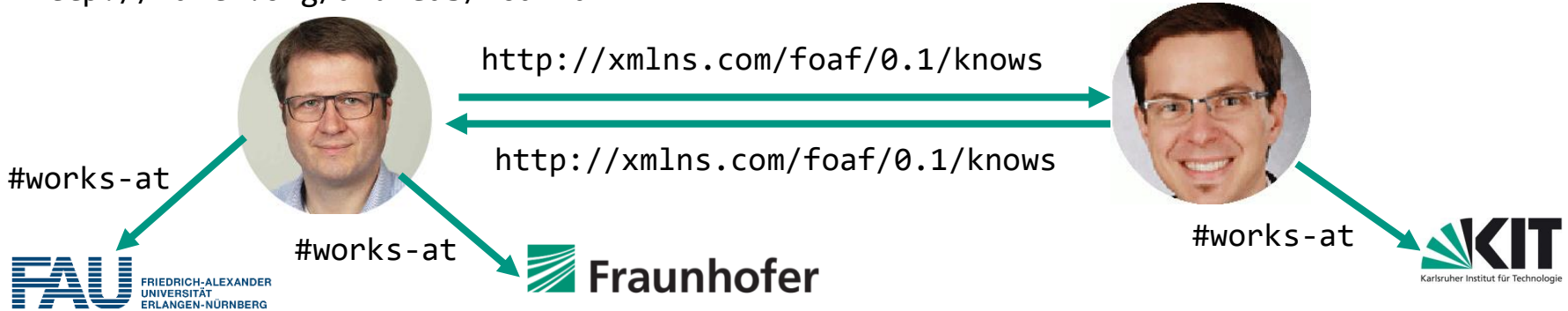
Tutorial on Distributed Knowledge Graphs for the Web of Things, Part I: Linked Data

Tobias Käfer (KIT) and Andreas Harth (FAU)

Tutorial @ 10th International Conference on the Internet of Things (IoT), 2020

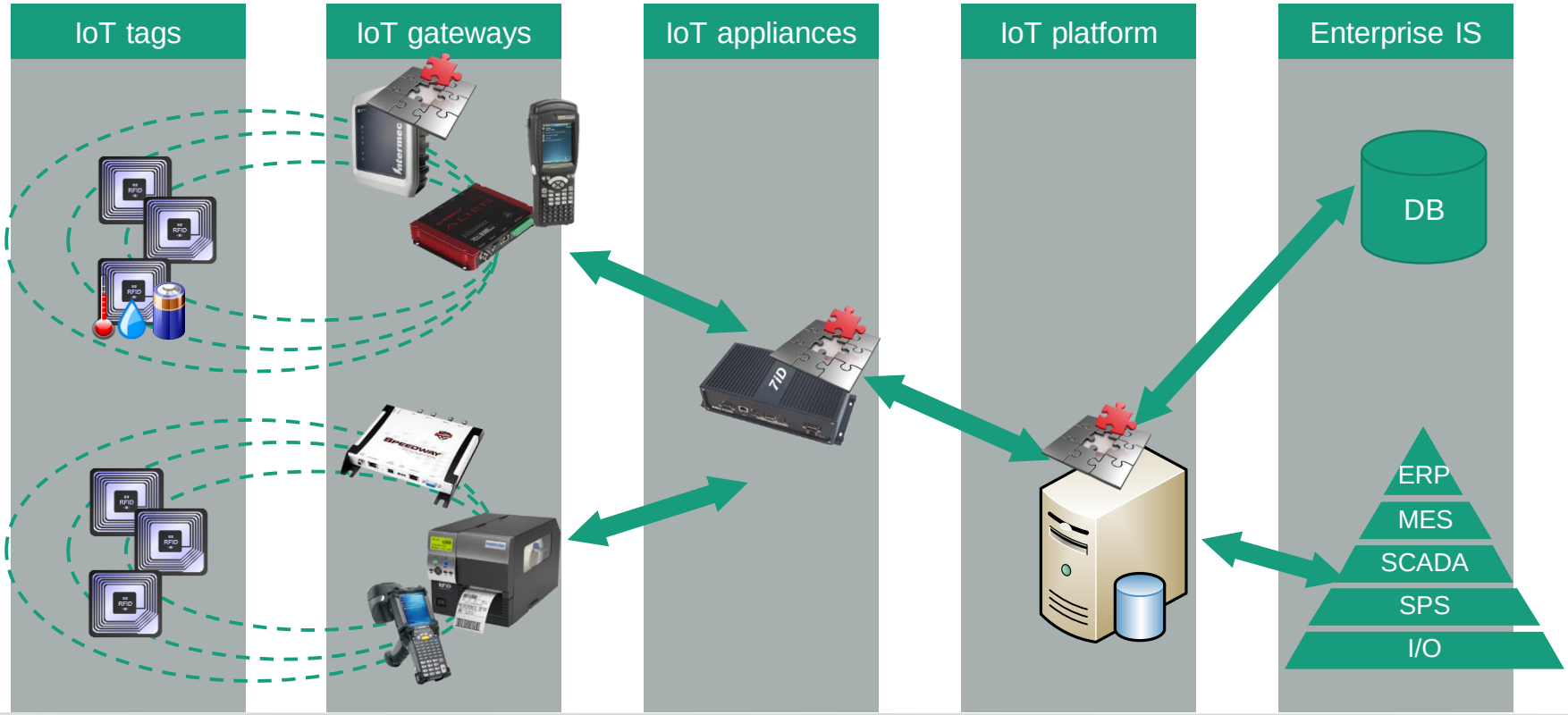


Agenda

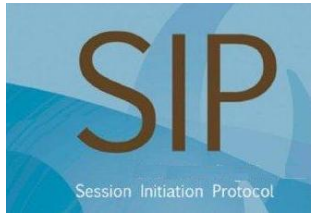
- **Introduction**
- Outline and Goal
- The Four Linked Data Principles
- Summary

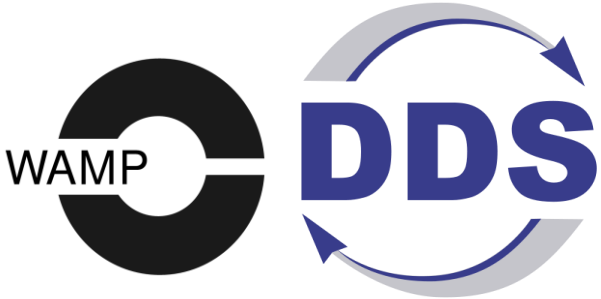
Lempert, S., 2015. *RFID-Middleware. Metastudie und Marktübersicht: Grundlagen, Anbieter und Produkte aus dem proprietären und Open Source-Umfeld*. Stuttgart: Fraunhofer Verlag. ISBN 978-3-8396-0915-6.

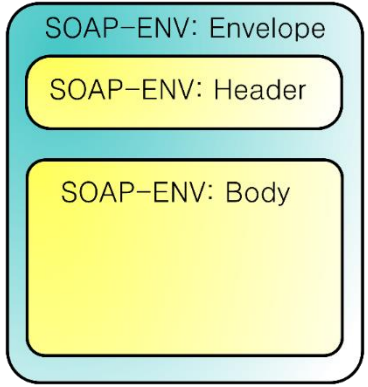
Classic IoT System Architecture



Which Network Protocol?





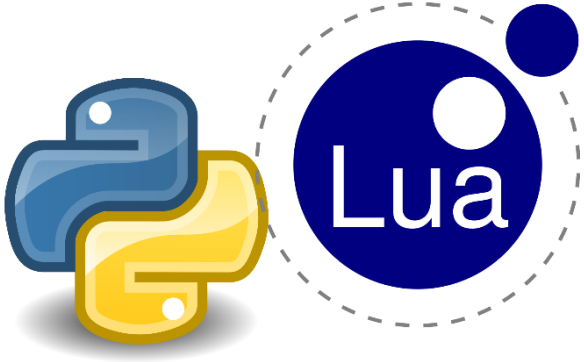


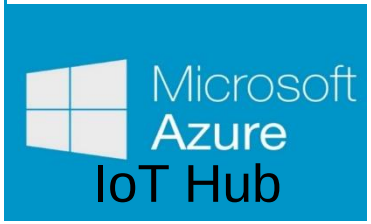
Which Data Model? Which Data Format?

And How to Specify and Run Applications?

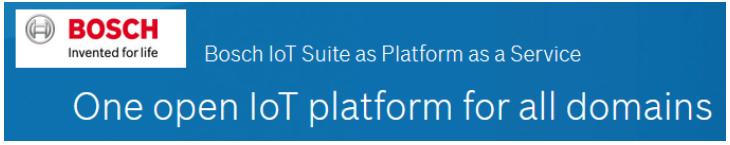
BPEL

BPMN





ASM

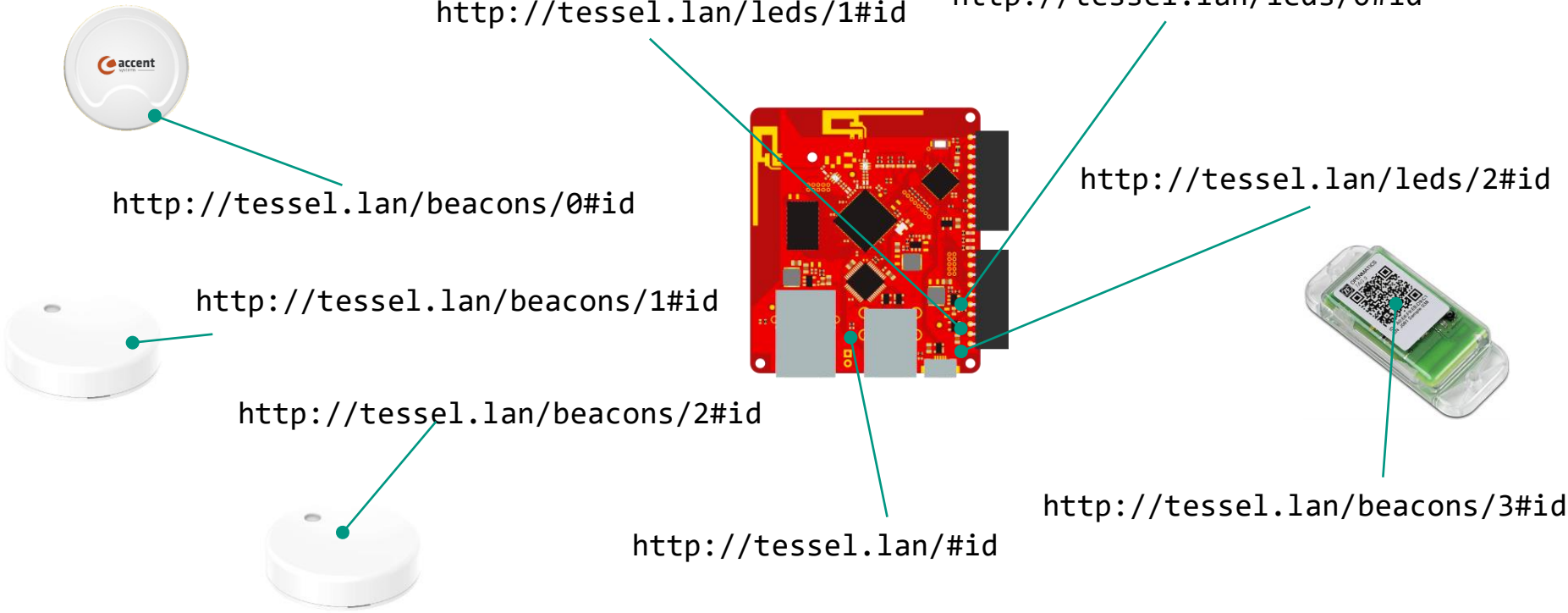


ECA

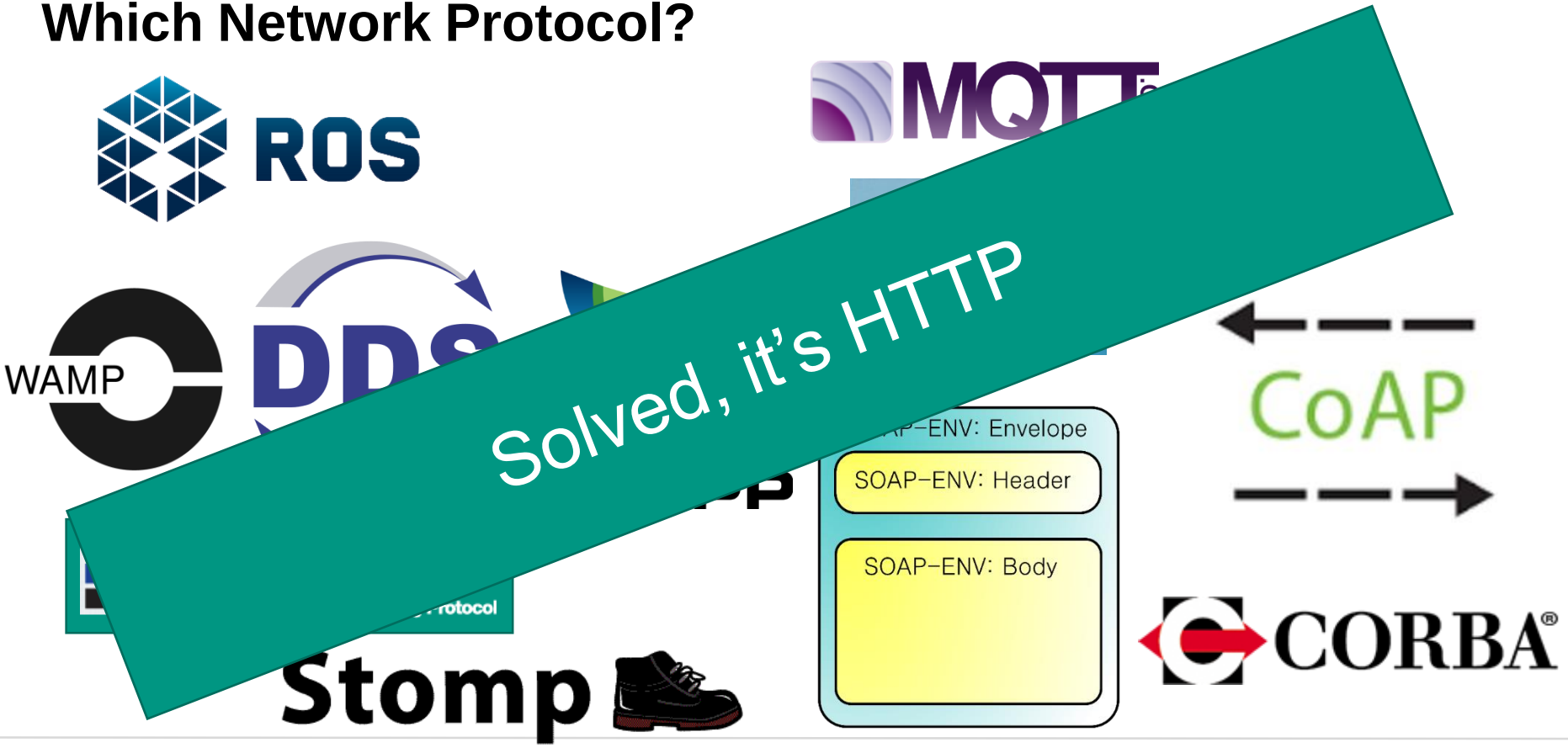
Agenda

- Introduction
- **Outline and Goal**
- The Four Linked Data Principles
- Summary

Basic Idea 1: HTTP URIs for Sensors and Actuators



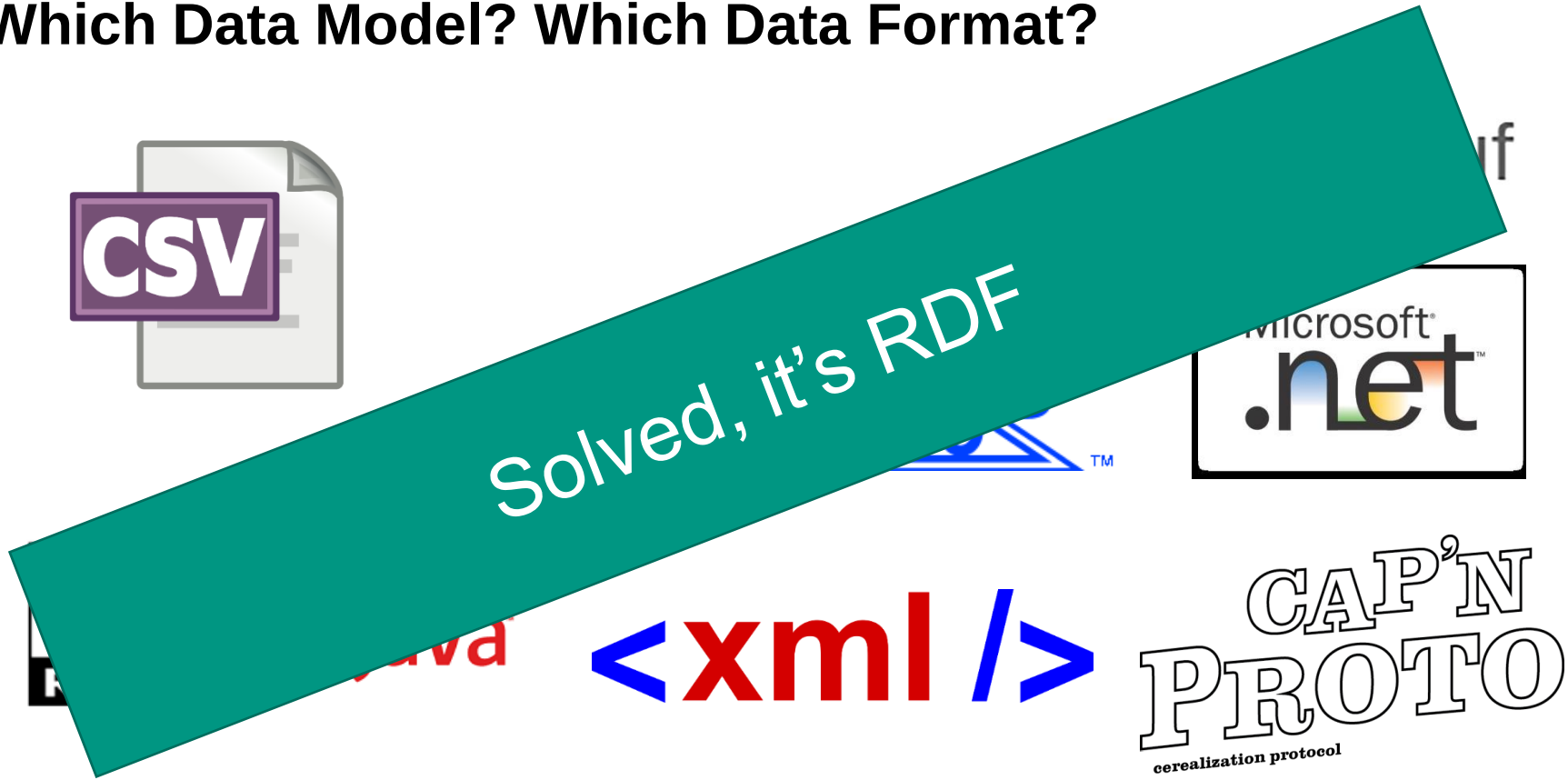
Which Network Protocol?



Basic Idea 2: Use Linked Data for Data Access

- HTTP for data access and manipulation
- Data providers are HTTP servers, data consumers are HTTP user agents
- Semantic Web languages (RDF, RDFS, a bit of OWL) for data representation and integration
- SPARQL for querying

Which Data Model? Which Data Format?



Basic Idea 3: Specify Application Behaviour Declaratively

- Using rules
  - IF condition THEN assertion (derivation rules)
  - IF condition THEN request (request rules, a variant of production rule)
- Using workflows (layered on top of the rules layer)
  - Using Sequence, Parallel, Conditional
- Run in decentralised settings

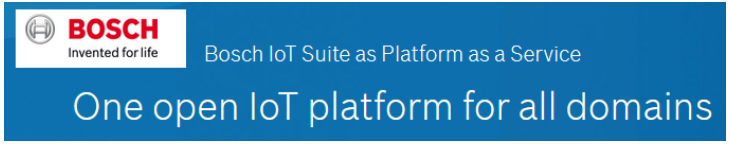
And How to Specify and Run Applications?

More research on rules and workflows
TBD

BPMN

loud Platform

ASM



ECA

Overall Goal of the Tutorial

“The tutorial covers web technologies (URIs, HTTP, RDF)
for specifying and executing applications (Recipes, Applets)
involving networked sensors and actuators (Internet of Things)
based on a logical representation of world state (knowledge representation)
and application behaviour.” (dynamical systems)

Goals of the Tutorial

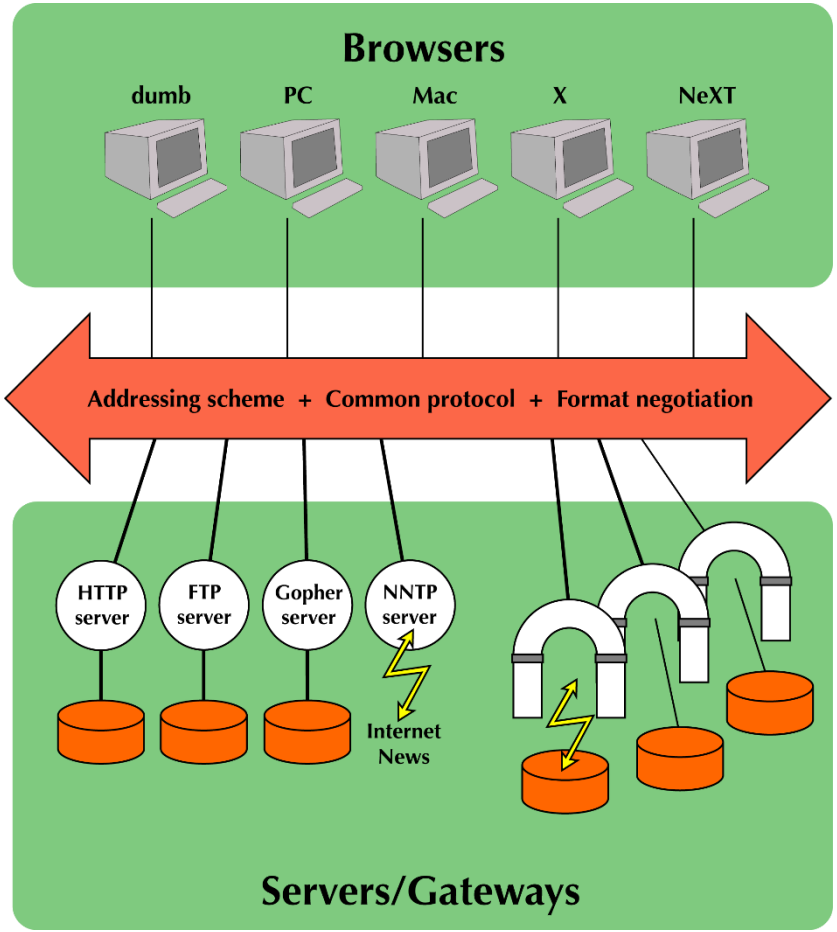
- We build on the basic notions of web architecture
  - HTTP URIs, Request/Response
  - Manipulation of resource state
  - Hyperlinks
- Our user agents operate on resources
  - Current resource state („now“), no distinction between „static“ and „dynamic“ data
- We show how to specify user agent behaviour
  - Declarative queries and rules, no imperative programming
  - Simple language can be basis for visual programming language
- Overall: we value simplicity to achieve web-scale
  - Focus on execution of application behaviour on standardised interfaces
  - No protocol mappings, not syntax mappings, no artificial intelligence planning

Agenda

- Introduction
- Outline and Goal
- **The Four Linked Data Principles**
- Summary

Web Architecture

- URI: RFC 1630 (1994), now RFC 3986
- HTTP: RFC 1945 (1996), now RFC 7230, 7231, 7232, 7234, 7235



https://www.w3.org/DesignIssues/diagrams/history/Architecture_crop.png
Redrawn from an image from 1990

Servers and User Agents

- Client

“A program that establishes connections for the purpose of sending requests.”

- User Agent

“The client which initiates a request. These are often browsers, editors, spiders (web-traversing robots), or other end user tools.”

- Server

“An application program that accepts connections in order to service requests by sending back responses.”

“Any given program may be capable of being both a client and a server; our use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general. […]”

All definitions from RFC2616 (obsolete)

Uniform Resource Identifiers (RFC 3986)

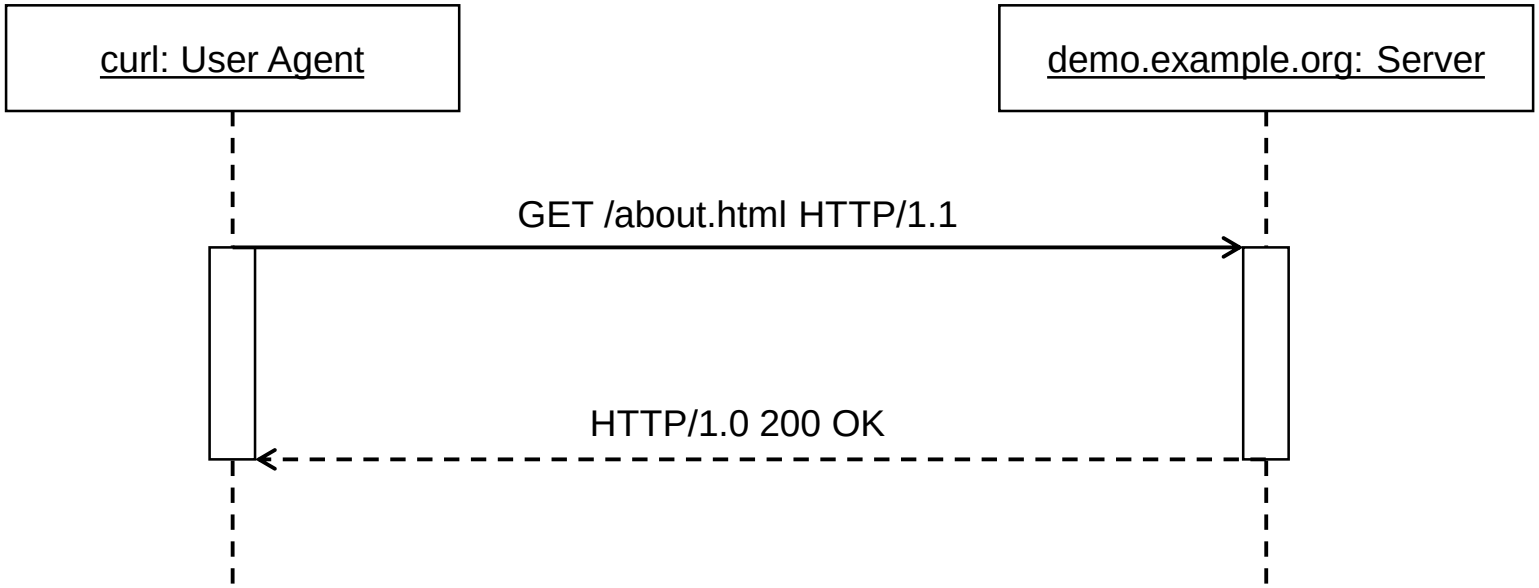
- Uniform
- Resource

“This specification does not limit the scope of what might be a resource; rather, the term ‘resource’ is used in a general sense for whatever might be identified by a URI.”

- an electronic document
- an image
- a source of information with a consistent purpose (e.g., ‘today's weather report for Los Angeles’)
- a service (e.g., an HTTP-to-SMS gateway)
- a collection of other resources
- the operators and operands of a mathematical equation
- the types of a relationship (e.g., ‘parent’ or ‘employee’)
- numeric values (e.g., zero, one, and infinity)

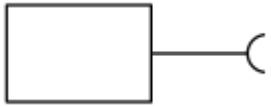
- Identifier

Request/Response in HTTP

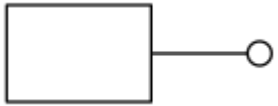


Request/Response and User Agent/Server

- REST assumes request/response communication pattern between components with client connector and server connector
  - Clients emit requests, receive response



  - Servers answer to incoming requests with a response



Linked Data

- Postulated by Tim Berners-Lee in 2006.

> *"The Semantic Web isn't just about putting data on the web. It is about making links, so that a person or machine can explore the web of data. With linked data, when you have some of it, you can find other, related, data."* [1]

- Collection of principles governing the publication and consumption of data on the web
- Aim: unified method for describing resources and accessing the state of resources
- Later we shall see how to manipulate (change) the state of resources

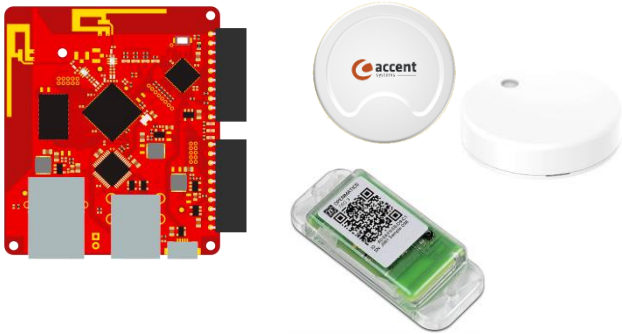
[1] http://www.w3.org/DesignIssues/LinkedData.html

Linked Data Principles

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
4. Include links to other URIs. so that they can discover more things.

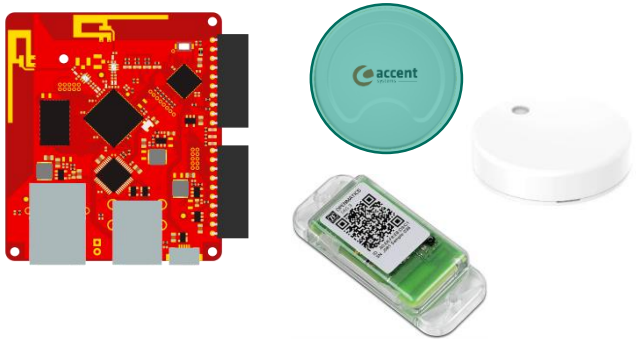
http://www.w3.org/DesignIssues/LinkedData.html

1 and # 2 Resources and URIs
Tessel and Beacons



Resource	URI
The Tessel device	`http://tessel.lan/#id`
The first beacon	`http://tessel.lan/beacons/0#id`
The second beacon	`http://tessel.lan/beacons/1#id`
The third beacon	`http://tessel.lan/beacons/2#id`
The fourth beacon	`http://tessel.lan/beacons/3#id`
The first LED	`http://tessel.lan/leds/0#id`
The second LED	`http://tessel.lan/leds/1#id`
The third LED	`http://tessel.lan/leds/2#id`

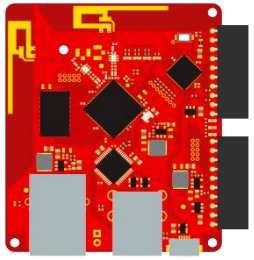
3: Provide Useful Information in RDF



- A User Agent performing a HTTP GET on `http://tessel.lan/beacons/0#id` leads to:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://example.org/woto#> .

</beacons/0#id> rdf:type :Sensor ;
                :mac “0x012345678” .
```

3: Provide Useful Information in RDF



- A User Agent performing a HTTP GET on `http://tessel.lan/beacons/3#id` leads to:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://example.org/woto#> .

</beacons/3#id> rdf:type :Sensor ;
                :hasProperty <#prop1> , <#prop2> .
<#prop1> rdf:type :ObservableProperty ;
       :measure :temperature ; :value "23" ; :unit "Celcius" .
<#prop2> rdf:type :ObservableProperty ;
       :measure :humidity ; :value "0.07" ; :unit “g/m3" .
```

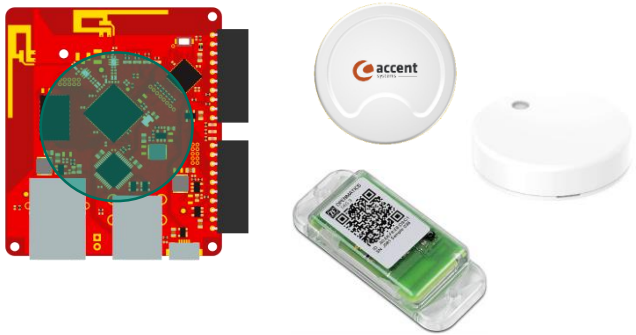
Resource Description Framework (RDF)

- A graph-structure data format (a formal language) for knowledge representation
- Subject-predicate-object triples
- URIs and literals to name resources; blank nodes as “variables”
- Graph structure enables merging of RDF graphs from multiple sources
- Different serialization syntaxes for the triple structure (e.g., XML, JSON)
- We use Turtle syntax to encode RDF triples and graphs
  - @prefix for compact URIs (CURIEs)
  - <> for URIs, e.g., <http://tessel.lan/beacons/0#id>
  - "" for literals, e.g., "23"
  - _: for blank nodes, e.g., _:bn
  - One triple per line, with . at the end
  - , to repeat subject and predicate, ; to repeat subject

RDF Abstract Syntax

Definition (RDF Terms, RDF Triple, RDF Graph) *The set of RDF terms consists of the set of URIs $\mathcal{U}$, the set of blank nodes $\mathcal{B}$ and the set of RDF literals $\mathcal{L}$, all being pairwise disjoint. A tuple $\langle s, p, o \rangle \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ is called an RDF triple, where s is the subject, p is the predicate and o is the object. A set of triples is called RDF graph.*

4: Links to other URIs



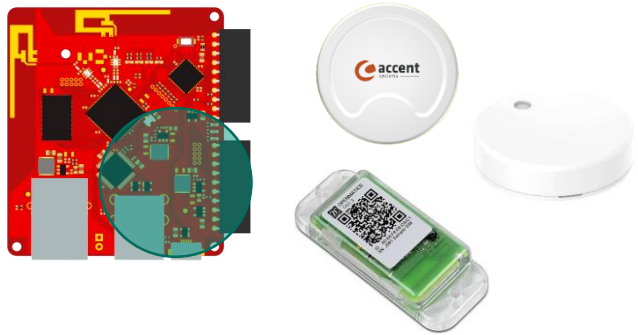
- A User Agent performing a HTTP GET on `http://tessel.lan/` leads to:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://example.org/woto#> .

<#id> rdf:type :Platform ; rdfs:comment "Andreas' Tessel2" ;
      :hosts <beacons/#id> , <leds/#id> .
```

- A HTTP GET on `http://tessel.lan/beacons/` leads to:

```
@prefix : <http://example.org/woto#> .

<#id> :hosts <0#id> , <1#id> , <2#id> , <3#id> .
```

Read-Write Linked Data



- A User Agent performing a HTTP PUT on `http://tessel.lan/leds/0` with the following:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://example.org/woto#> .

</leds/0#id> rdf:type :Actuator ;
             :hasProperty <#prop1> .
<#prop1> rdf:type :ActuatableProperty ;
       :onOffStatus :on .
```

Agenda

- Introduction
- Outline and Goal
- The Four Linked Data Principles
- **Summary**

Summary

- Commercial IoT platforms aim at hiding differences in communication protocols and data representation
- Web technologies offer a vendor-neutral path to accessing sensor data and effecting change in actuators

- A uniform communication protocol helps when accessing different sensors and actuators
- Uniform data representation helps when reading sensor state (later: writing actuator state)

- Linked Data builds on web architecture (URIs, HTTP) and graph-structured data (RDF)
- URIs and HTTP are the basis for data access and manipulation
- RDF is the basis for knowledge representation and integration of various message formats (plus RDFS and a bit of OWL)

- Linked Data uses a decentralised architecture (no centralised registry or catalogue), discovery via hyperlinks